Using NGiNX for release automation at The Atlantic

The Atlantic

About the speakers

Mike Howsden is the DevOps Lead at *The Atlantic*

Frankie Dintino is a Senior Full-Stack Developer at The Atlantic

One Virtual Machine

Nearly Unlimited QA environments

NGINX

The Atlantic

Our development workflow

- Two week sprints
- New code is committed to a branch in git
- When coding is complete, a Pull Request is opened in GitHub for review
- Branch can be staged onto a beta server for QA
- PR is merged into a branch named `develop` when code review and QA is complete
- The `develop` branch is merged into the `master` branch when it's time for a deployment

Discontinuous Integration (the old way)

```
ssh betaserver
cd /www/beta5
source bin/activate
git fetch origin && git reset --hard && git checkout -t origin/my-branch
pip install --upgrade -r requirements.txt
(cd frontend && npm install && gulp build)
importdb atl_db_beta5
django-admin collectstatic -l --noinput
touch wsgi.py
ohpleasegod --help
```

Discontinuous Integration

Why was this bad?

- **Tedious!**

Discontinuous Integration

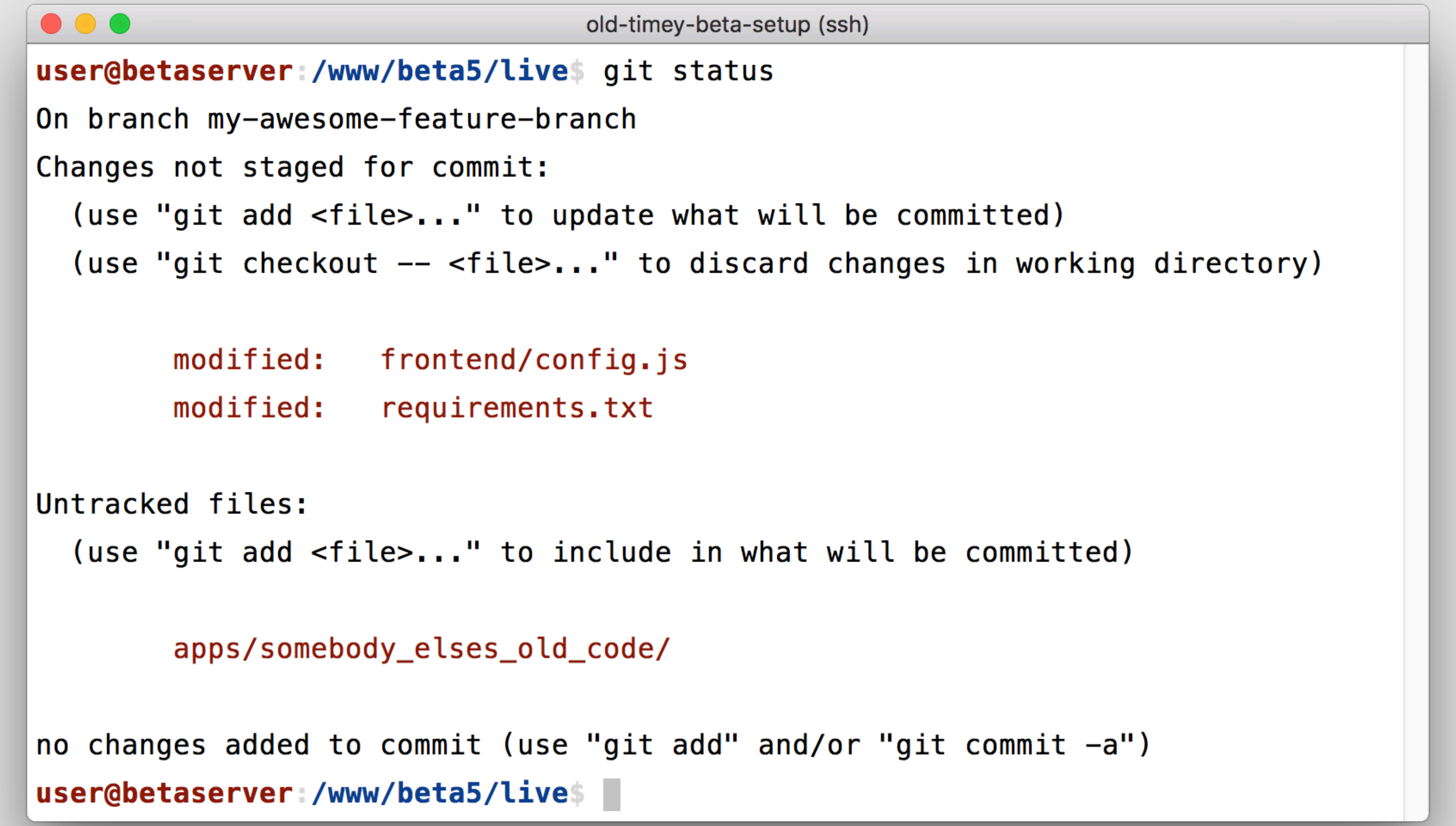
Why was this bad?

- Tedious!
- **Error prone!**

Discontinuous Integration

Why was this bad?

- Tedious!
- Error prone!
- **Stateful!**


old-timey-beta-setup (ssh)

```
user@betaserver:/www/beta5/live$ git status
On branch my-awesome-feature-branch
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   frontend/config.js
        modified:   requirements.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        apps/somebody_elses_old_code/

no changes added to commit (use "git add" and/or "git commit -a")
user@betaserver:/www/beta5/live$
```

The Atlantic

Discontinuous Integration

Why was this bad?

- Tedious!
- Error prone!
- Stateful!
- **Enough instances to waste resources, not enough to ensure one is always available**

Discontinuous Integration

Why was this bad?

- Tedious!
- Error prone!
- Stateful!
- Enough instances to waste resources, not enough to ensure one is always available
- **Worst of all, NGINX played only a minor role**

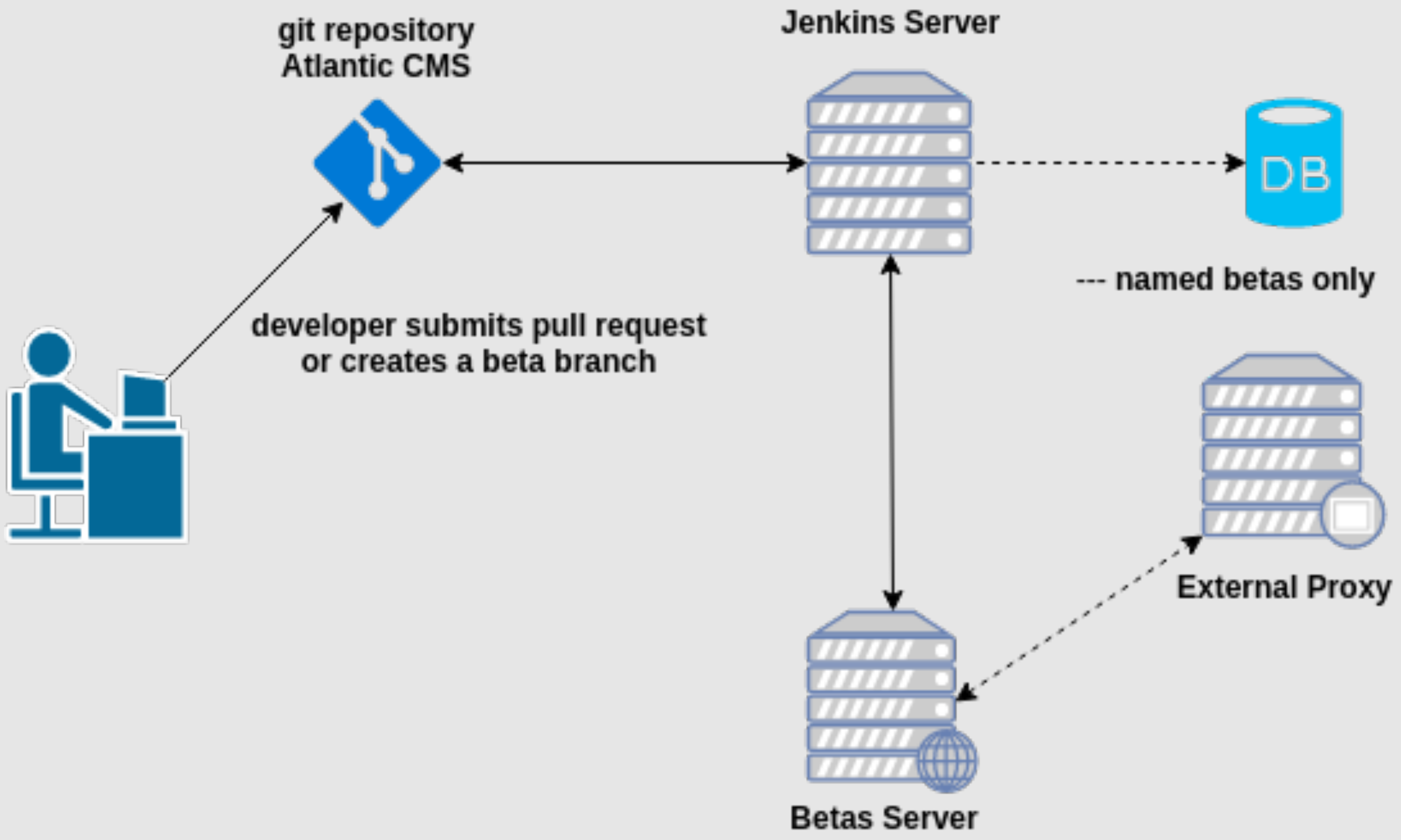
We can do better!

The Atlantic

Business Need

- Developers need to easily stage their work for peer and stakeholder review.
-

Beta architecture overview



Infinite* beta environments—components

- Jenkins
- Reflinks
- supervisord managing uWSGI emperor mode
- nginx-mod-passenger for NodeJS apps
- NGiNX and uWSGI socket files
- Database snapshots
- External proxy

* ish—as many as we need

The Atlantic

New process

Two common use-cases, with different requirements:

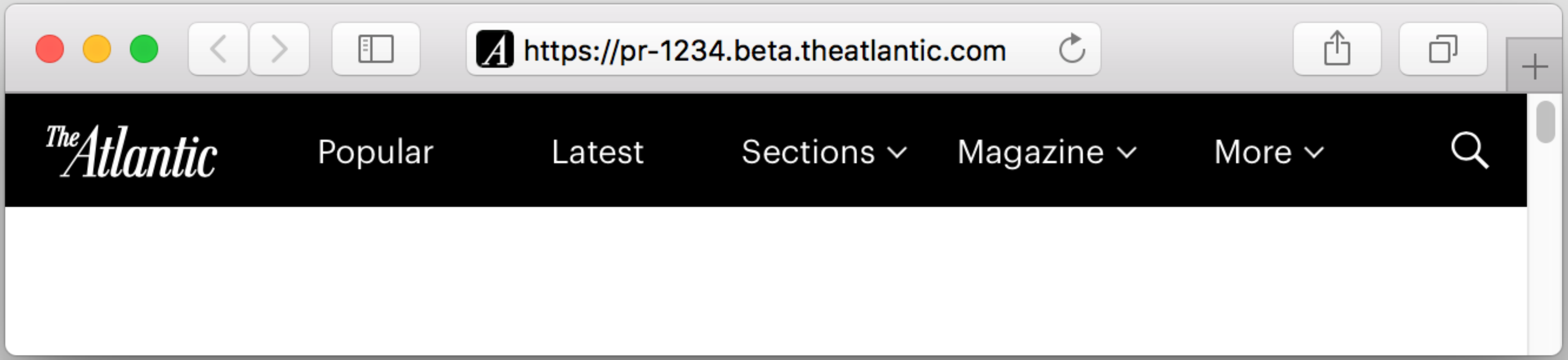
1. Simple bug fixes and minor features
2. Long-running feature branches

New process—typical setup

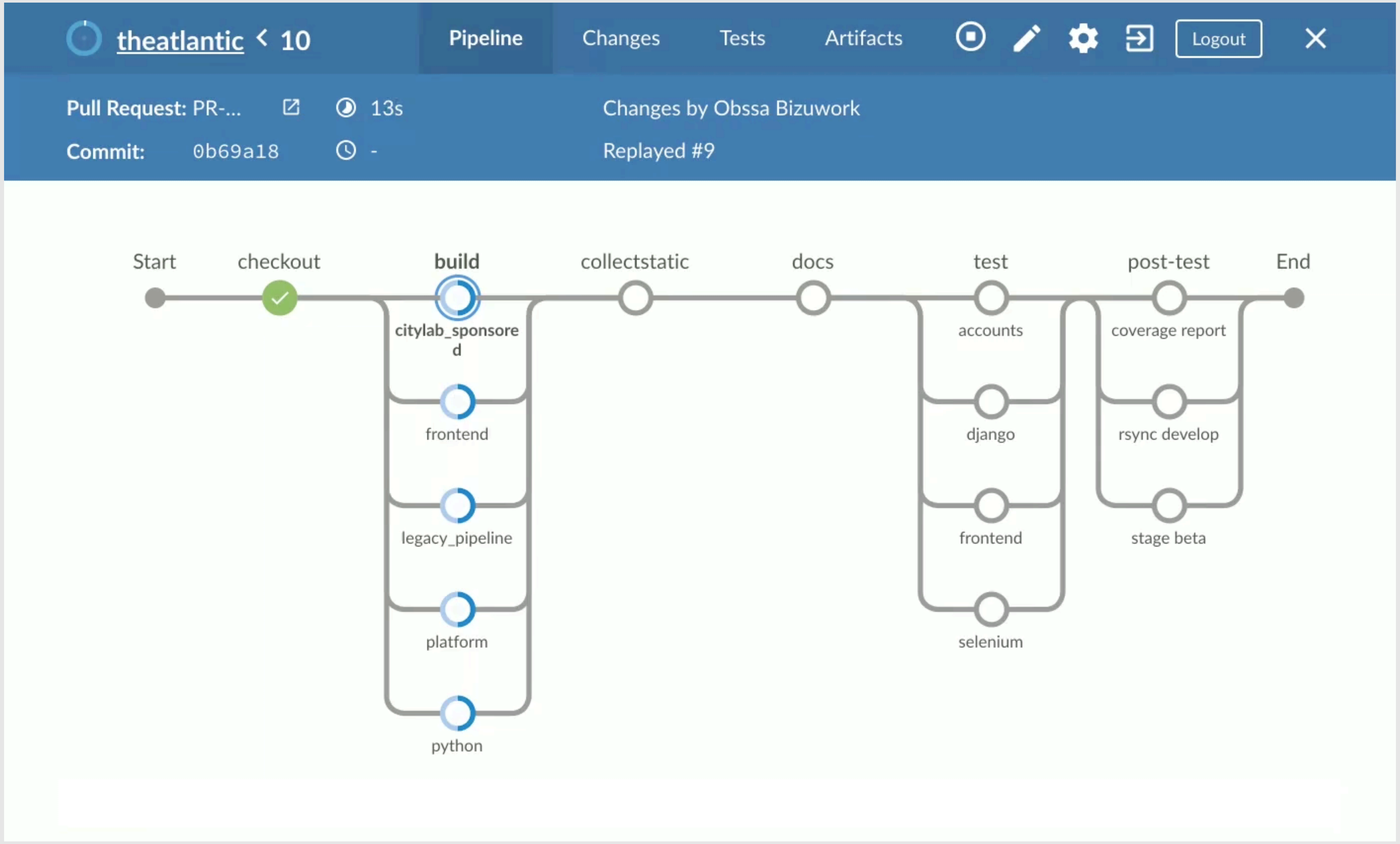
1 Open a pull request

2 Jenkins build (roughly 10 minutes)

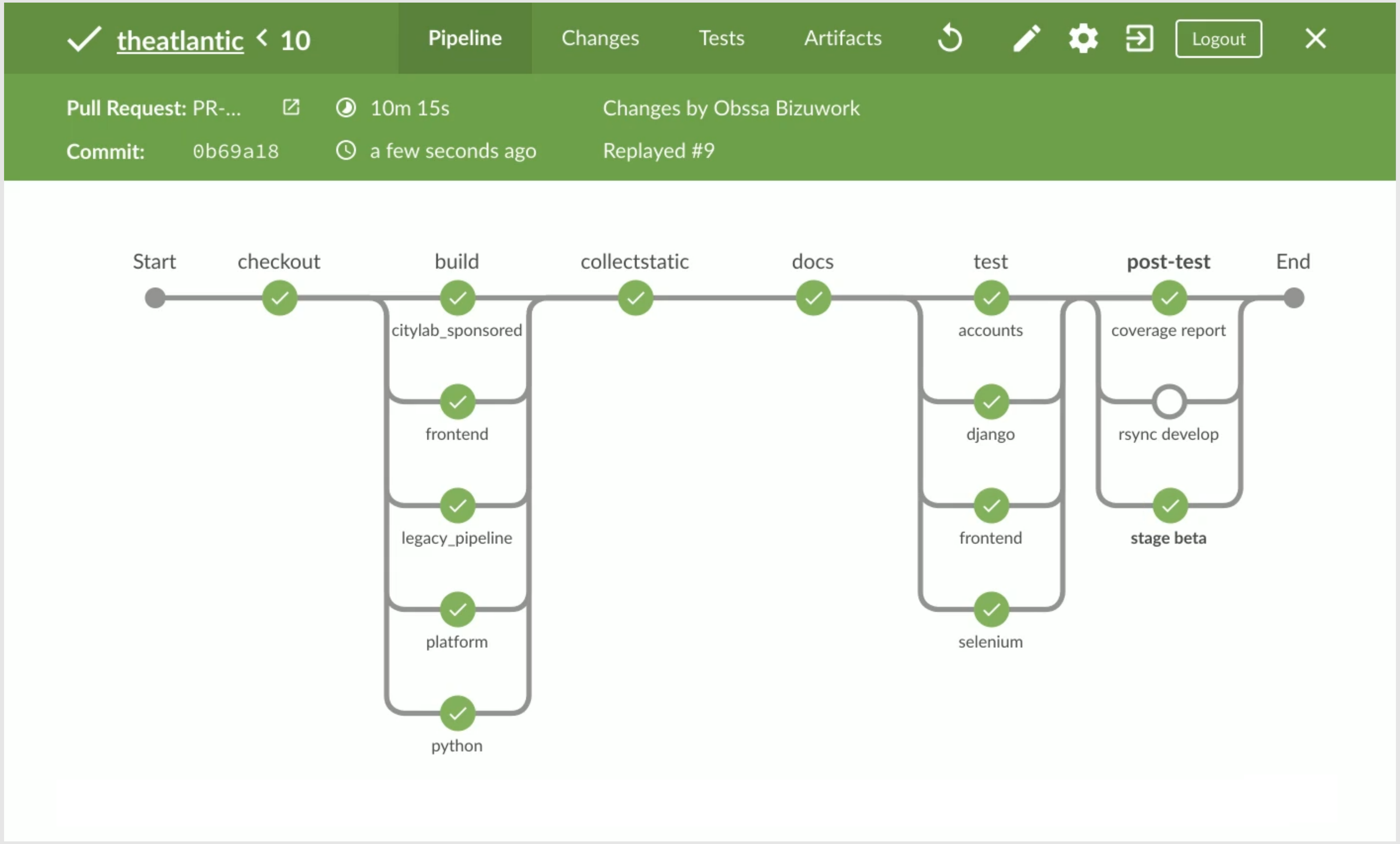
3



The Atlantic



Pull Request: PR-...
13s
Changes by Obssa Bizuwork
Commit: 0b69a18
-
Replayed #9


theatlantic < 10
Pipeline
Changes
Tests
Artifacts
Logout
Pull Request: PR-...
10m 15s
Changes by Obssa Bizuwork
Commit: 0b69a18
a few seconds ago
Replayed #9
Start
checkout
build
citylab_sponsored
frontend
legacy_pipeline
platform
python
collectstatic
docs
test
accounts
django
frontend
selenium
post-test
coverage report
rsync develop
stage beta
End

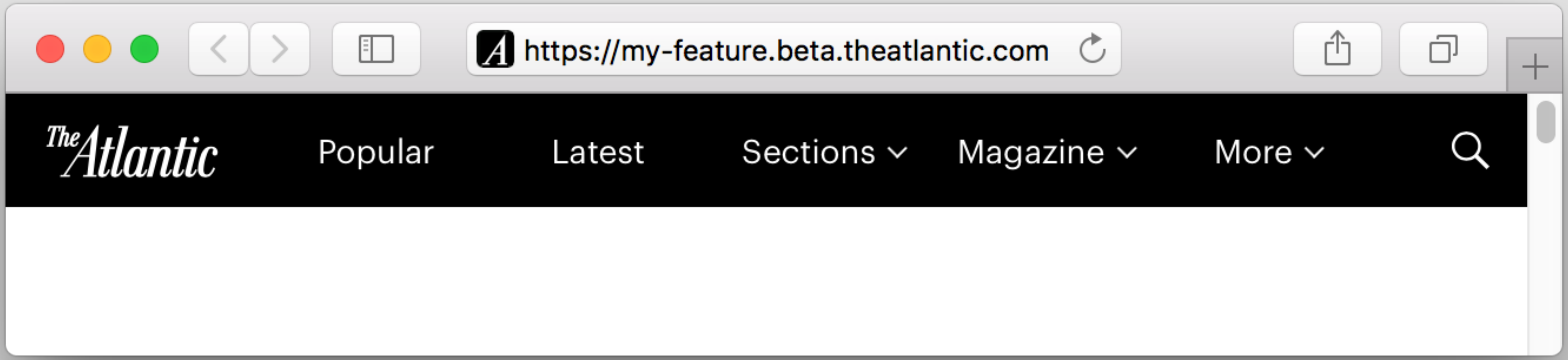
New process—"named betas"

- Uses a distinct, persistent copy of the database (to allow for schema changes and feature-specific data)
- Accessible outside the internal network with authentication

New process—"named betas"

1. Push out a branch: `beta/my-feature`

2. Jenkins build (roughly 10 minutes)

3. 

The Atlantic

Business Value

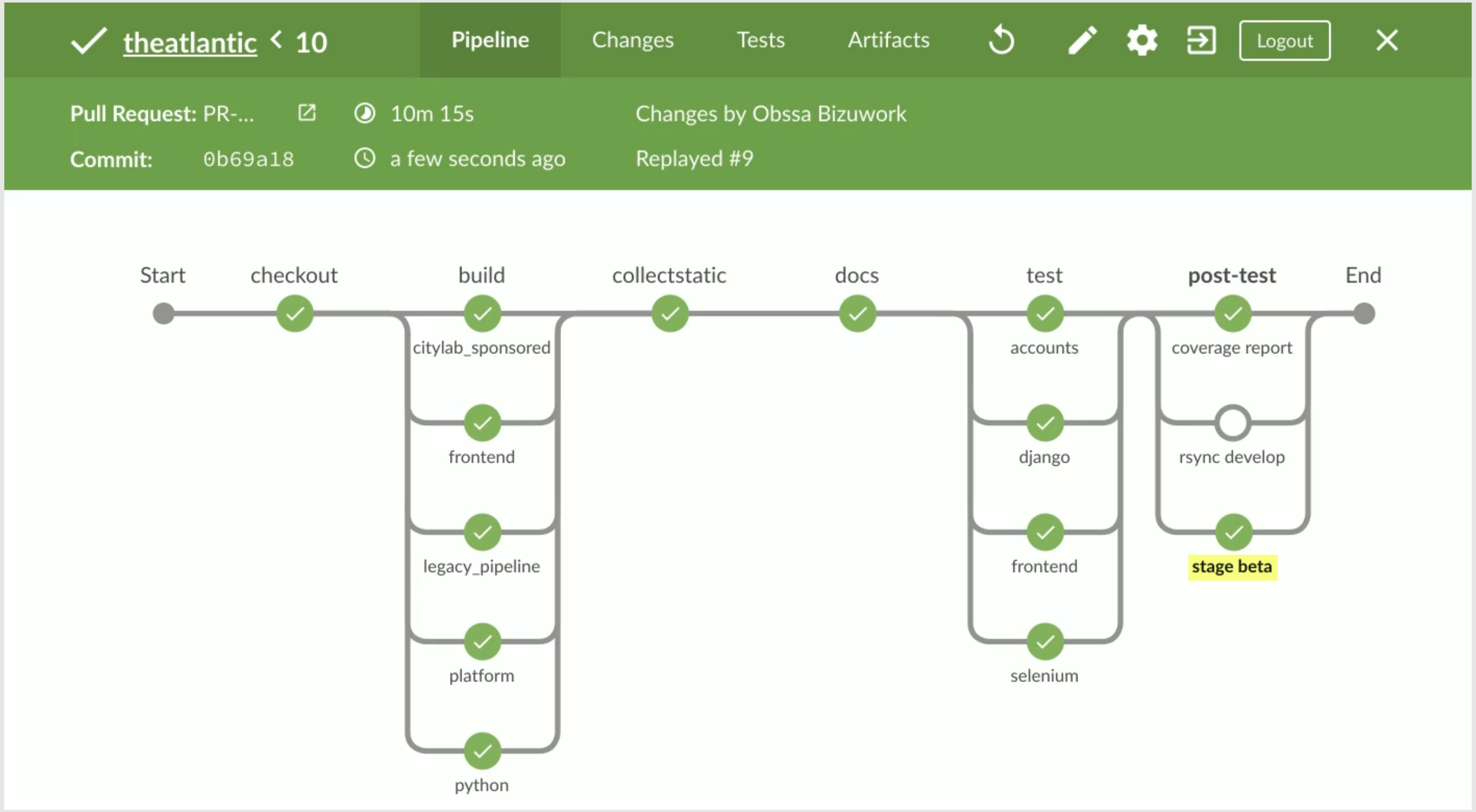
- **Efficiency**: Cuts down on developer time spent on tasks unrelated to business goals.
- **Morale**: Automates repetitive/monotonous work.
- **Improves Coverage/Consistency**: No barriers to manual testing, even for trivial changes. Each build environment is created with the exact same process.

Infinite* beta environments—components

- **Jenkins**
- Reflinks
- supervisord managing uWSGI emperor mode
- nginx-mod-passenger for NodeJS apps
- NGiNX and uWSGI socket files
- Database snapshots
- External proxy

* ish—as many as we need

Jenkins—stage beta



Pull Request: PR-...

Commit: 0b69a18

10m 15s

a few seconds ago

Changes by Obssa Bizuwork

Replayed #9

Jenkinsfile

- A declarative build and deployment script, committed to source control in the root of the repo
- Hooked into github notifications for pull request opens and branch pushes
- Reasonably powerful control over conditional logic, parallelization, pass/fail conditions, etc.

Jenkinsfile

```
pipeline {
  // ...
  stage('stage beta') {
    when { anyOf {
      expression { BRANCH_NAME ==~ /PR-[0-9]+/ }     // a pull request
      branch 'develop'                               // development branch
      expression { BRANCH_NAME.startsWith('beta/') } // 'beta/feature'
    } }
    steps {
      // Initialize database
      sh """ ssh devdbserver "create_beta_db $BRANCH_NAME" """
      // If the workspace doesn't exist, create a reflink copy
      sh """ssh betaserver \\
              "[ -d ${WORKSPACE} ] || cp -ar --reflink=auto \\
              /www/builds/theatlantic/develop.base ${WORKSPACE}" """
      // ...continued
```

Jenkinsfile (continued)

```
// ...
 stage('stage beta') {
   // ...
   steps {
     // Copy current build to remote workspace dir
     sh """rsync -acvzh0 --delete \\
             --exclude "*.pyc" --exclude .git --exclude ... \\
             $WORKSPACE/ betaserver:$WORKSPACE/ """

     // Create wsgi files, which will initialize uWSGI emperor processes
     sh """ssh betaserver "$WORKSPACE/support/beta/create_wsgi.py" """
   }
 }
```

The Atlantic

Infinite* beta environments—components

- Jenkins
- **Reflinks**
- supervisord managing uWSGI emperor mode
- nginx-mod-passenger for NodeJS apps
- NGiNX and uWSGI socket files
- Database snapshots
- External proxy

* ish—as many as we need

The Atlantic

Jenkins—stage beta

- Initializes with a reflink copy of a previous build to save on disk space (1.2G => 3M)
- Uses rsync to copy the newly created environment to the betas server over top of this copy

Reflinks (copy-on-write)

- BTRFS on debian or ubuntu
- XFS on RHEL, CentOS, or Fedora
  - `mkfs.xfs` must be called with `reflink=1`
  - `cp -ar --reflink=auto`
  - only uses the space required for differences between builds
  - considered “experimental”—requires custom-built kernel (may be enabled by default in Fedora 29)

Infinite* beta environments—components

- Jenkins
- Reflinks
- **supervisord managing uWSGI emperor mode**
- nginx-mod-passenger for NodeJS apps
- NGiNX and uWSGI socket files
- Database snapshots
- External proxy

* ish—as many as we need

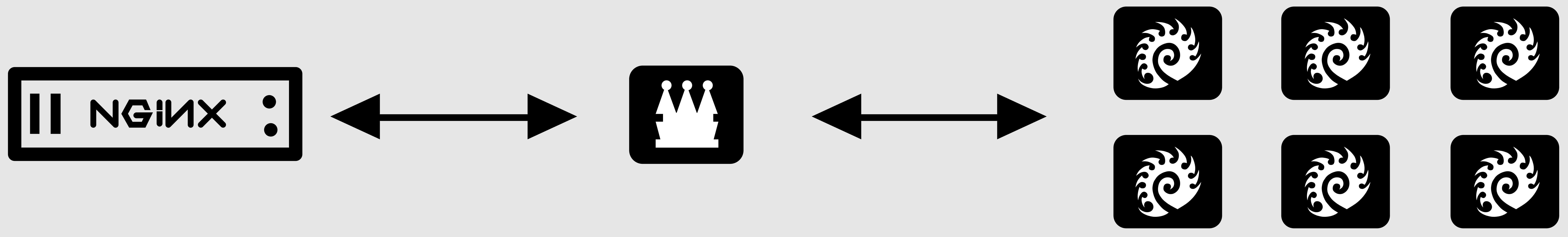
uWSGI emperor + supervisord

```
[program:zerg_server]
command = /sbin/uwsgi
  --master --thunder-lock
  --emperor "/www/builds/theatlantic/*/uwsgi/master.*.ini"
autorestart = true

[program:workers]
command = /sbin/uwsgi
  --master --thunder-lock
  --emperor "/www/builds/theatlantic/*/uwsgi/workers.*.ini"
autorestart = true
```

The Atlantic

Infinite* beta environments—components



uWSGI zerg master

uWSGI zerg workers

uWSGI emperor + supervisord

```
; /www/builds/theatlantic/pr-1234/uwsgi/master.theatlantic.ini
[uwsgi]
; %n: current config filename, without the .ini extension
; Strip off the leading 'master.' part of the filename to determine the app.
app = @(exec:///usr/bin/perl -e '$_ = "%n"; s/^master\.//g; print')

subdomain = %3 ; the 4th path component (0-indexed); eg pr-1234

thunder-lock = true
master = true
processes = 0

http = /var/run/uwsgi/%(app)-%(subdomain).sock
zerg-server = /var/run/uwsgi/%(app)-%(subdomain)-workers.sock
```

uWSGI zerg server

```
; /www/builds/theatlantic/pr-1234/uwsgi/master.theatlantic.ini
[uwsgi]
; %n: current config filename, without the .ini extension
; Strip off the leading 'master.' part of the filename to determine the app.
app = @(exec:///usr/bin/perl -e '$_ = "%n"; s/^master\.//g; print')

subdomain = %3 ; the 4th path component (0-indexed); eg pr-1234

thunder-lock = true
master = true
processes = 0

http = /var/run/uwsgi/%(app)-%(subdomain).sock
zerg-server = /var/run/uwsgi/%(app)-%(subdomain)-workers.sock
```

The Atlantic

uWSGI zerg workers

```ini
; /www/builds/theatlantic/pr-1234/uwsgi/worker.theatlantic.ini
[uwsgi]
; attach to zerg pool
zerg = /var/run/uwsgi/%(app)-%(subdomain)-workers.sock

cheaper-algo = busyness
cheaper = 1               ; minimum number of workers to keep at all times
cheaper-initial = 2       ; starts with minimal workers
cheaper-step = 2          ; spawn at most 2 workers at a time
cheaper-overload = 30     ; seconds between busyness checks
cheaper-busyness-multiplier = 50
cheaper-busyness-penalty = 2

; how many requests are in backlog before quick response triggered
cheaper-busyness-backlog-alert = 33
cheaper-busyness-backlog-step = 1
idle = 86400 ; workers shut down if the beta is idle for a day

wsgi-file = %(env_dir)/apps/wsgi.py
env = DJANGO_SETTINGS_MODULE=settings.%(app).live
```

The Atlantic

NGiNX, proxying to uWSGI socket files

```
server {
    server_name "~^(?<subdomain>.+)\.beta\.theatlantic\.com$";
    root /www/builds/theatlantic/$subdomain;

    location / {
        try_files /assets/$uri @django;
    }

    location @django {
        internal;
        include includes/proxypass.conf;
        proxy_pass http://unix:/var/run/uwsgi/theatlantic-$subdomain.sock;
    }
}
```

The Atlantic

nginx-mod-passenger for nodejs

```
server {
    server_name "~^(?<subdomain>.+)\.ampbeta\.theatlantic\.com$";

    root /www/builds/amp/${subdomain}/public;

    passenger_enabled on;
    passenger_app_type node;
    passenger_app_root /www/builds/amp/${subdomain};
    passenger_restart_dir /www/builds/amp/${subdomain};
    passenger_startup_file dist/host/bin/www.js;
}
```

Questions?

The Atlantic

Thank you!

github.com/theatlantic/nginxconf-2018

frankie@theatlantic.com
mhowsden@theatlantic.com